





顺序寻址

指令寻址

跳跃寻址

指令寻址: 如何确定下一条指令的存放地址?

一条指令的结构

操作码(OP)

地址码(可能有多个)

# 回忆: 计算机的工作过程

```
int a=2,b=3,c=1,y=0;
void main(){
   y=a*b+c;
}
```

#### 程序计数器 PC: 指明下一条指 令的存放地址

PC

下一条指令的地址:

$$(PC) + 1 \longrightarrow PC$$



按字节编址怎么办? 采用变长指令字结构怎么办?

				AZ Z . :
	主存	指令		CO NATES
	地址	操作码	地址码	注释
	0	000001	0000000101	取数a至ACC
	1	000100	0000000110	乘b得ab,存于ACC中
	2	000011	0000000111	加c得ab+c,存于ACC中
	3	000010	0000001000	将ab+c,存于主存单元
	4	000110	0000000000	停机
	5	5 000000000000000000000000000000000000		原始数据a=2
	6			原始数据 <b>b</b> =3
	7			原始数据 <i>c=1</i>
	8			原始数据y=0
AN	25-			

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

 $(PC) + 1 \longrightarrow PC$ 

#### 指令地址

0	0001001111101000
1	0011001111101001
2	0010010010110000
3	100100000000111
4	0001011111010000
5	0100011111010001
6	0101011111010001
7	0001100111000100
8	

该系统采用<mark>定长指令字结构</mark>

指令字长=存储字长=16bit=2B

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

 $(PC) + 1 \longrightarrow PC$ 

PC 0 +1

指令地址 操	作码 地址码
--------	--------

0

5

6

8

	LDA	1000
	ADD	1001
	DEC	1200
	JMP	7
	LDA	2000
	SUB	2001
	INC	)
Á	LDA	1100
X	3	

该系统采用<mark>定长指令字结构</mark>

指令字长=存储字长=16bit=2B

0

5

6

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

 $(PC) + 1 \longrightarrow PC$ 

PC 1 +1

指令地址	操作码	地址码
<b>7</b>	45 1 - 1 1 . 4	

LDA	1000
ADD	1001
DEC	1200
JMP	7
LDA	2000
SUB	2001
INC	
LDA	1100
	••

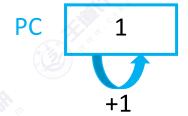
该系统采用<mark>定长指令字结构</mark>

指令字长=存储字长=16bit=2B

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

$$(PC) + 1 \longrightarrow PC$$



	4月. <i>16</i> . 77	しんしき ナロ
指令地址	操作码	地址码
		アビンアル、ドリ

LDA	1000
ADD	1001
DEC	1200
JMP	7
LDA	2000
SUB	2001
INC	
LDA	1100
	••

#### 该系统采用定长指令字结构

指令字长=存储字长=16bit=2B

主存<mark>按字编址</mark>

5

6

8

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

 $(PC) + 1 \longrightarrow PC$ 

PC 2 +1

指令地址	操作码	地址码
1月 文地址	1米1下1円	地址洞

0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7.
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8		

#### 该系统采用<mark>定长指令字结构</mark>

指令字长=存储字长=16bit=2B

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

$$(PC) + 1 \longrightarrow PC$$

$$(PC) + 2 \longrightarrow PC$$

#### 指令地址

0	0001001111101000
2	0011001111101001
4	0010010010110000
6	100100000000111
8	0001011111010000
10	0100011111010001
12	0101011111010001
14	0001100111000100

该系统采用定长指令字结构

指令字长=存储字长=16bit=2B

主存按<mark>字节</mark>编址

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

读入一个字,根据操作码判断这条指令的总字节数 n, 修改PC的值

$$(PC) + n \longrightarrow PC$$

根据指令的类型,CPU可能还要进行多次访存,每次读入一个字

#### 指令地址

	•••
	30
14	0001100111000100
12	0101011111010001
10	0100011111010001
8	0001011111010000
6	100100000000111
4	0010010010110000
2	0011001111101001
O	0001001111101000

0001001111101000

该系统采用<mark>变长</mark>指令字结构 <del>指令字长=存储字长=16bit=2B</del>

主存按字节编址

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

<mark>顺序寻址</mark> (PC)+"1" → PC

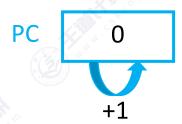
这里的1理解为1个指令字长,实际加的值会因指令长度、编址方式而不同

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

顺序寻址 (PC)+"1" → PC

跳跃寻址 由转移指令指出



指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8		

该系统采用定长指令字结构

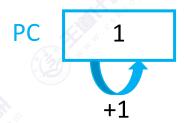
指令字长=存储字长=16bit=2B

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

顺序寻址 (PC)+"1" → PC

跳跃寻址 由转移指令指出



指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8		

该系统采用定长指令字结构

指令字长=存储字长=16bit=2B

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

当前正在执 指令地址 操作码 地址码 顺序寻址  $(PC) + "1" \longrightarrow PC$ 1000 LDA 由转移指令指出 跳跃寻址 1001 ADD DEC 1200 PC 3 **JMP** 2000 4 LDA 5 **SUB** 2001 6 INC LDA 1100 8 • • •

行的指令

指令字长=存储字长=16bit=2B

该系统采用定长指令字结构

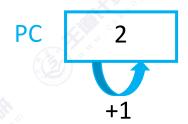
主存按字编址

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

顺序寻址 (PC)+"1" → PC

跳跃寻址 由转移指令指出



指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8		

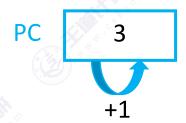
该系统采用定长指令字结构 指令字长=存储字长=16bit=2B

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

顺序寻址 (PC)+"1" → PC

跳跃寻址 由转移指令指出



指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7.7
4	LDA	2000
5	SUB 2001	
6	INC	
7	LDA	1100
8		

该系统采用<mark>定长指令字结构</mark>

指令字长=存储字长=16bit=2B

主存按字编址

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

顺序寻址 (PC)+"1" → PC

跳跃寻址 由转移指令指出

安次取指令之后, PC一定会自动+1, 指向下一条应该执 行的指令

指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	707
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8		<b></b>

该系统采用定长指令字结构

指令字长=存储字长=16bit=2B

主存按字编址

JMP: 无条件转移 把PC中的内容改成7

> 无条件转移指令, 类似C语言的 goto

操作码

指令寻址 下一条 欲执行 指令 的 地址

(始终由程序计数器PC给出)

顺序寻址 (PC)+"1" → PC 跳跃寻址 由转移指令指出

PC

执行转移指令,将 PC值修改为7 指令地址 0 顺序寻址 1 顺序寻址 2

顺序寻址 3 4 5

跳跃寻址 7

6

LDA	1000
ADD	1001
DEC	1200
JMP	7.
LDA	2000
SUB	2001
INC	
LDA	1100
400	

地址码

该系统采用定长指令字结构

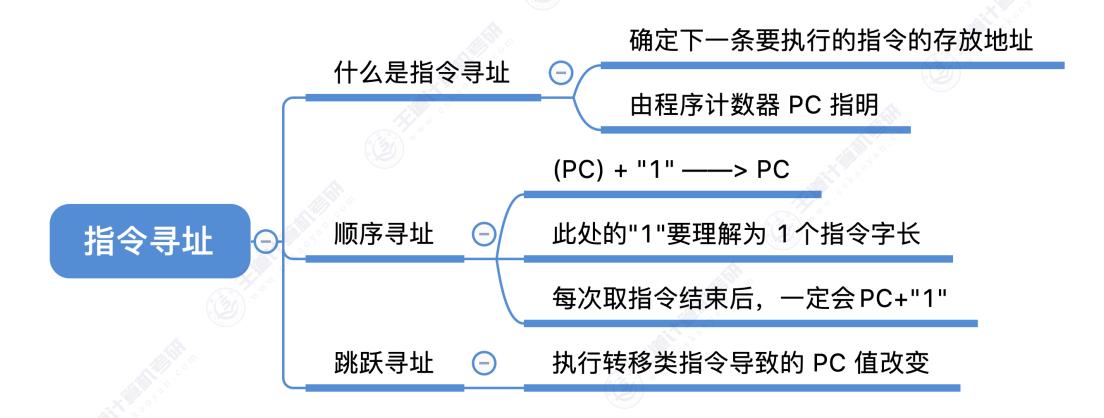
指令字长=存储字长=16bit=2B

主存按字编址

JMP: 无条件转移 把PC中的内容改成7

> 无条件转移指令, 类似C语言的 goto

#### 本节回顾



注:每一条指令的执行都分为"取指令"、"执行指令"两个阶段



公 公众号: 王道在线



b站: 王道计算机教育



**小** 抖音: 王道计算机考研